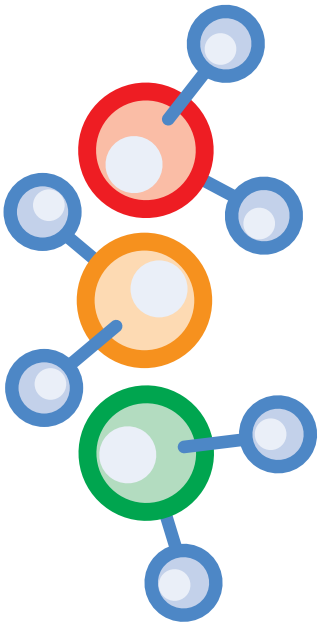

feu tricolore pour CO_2

connecté à Sigfox

Pas besoin de réseau WiFi !

Peter Groppe, Frank Schleking et Bernd vom Berg (Allemagne)

Nous avons déjà présenté plusieurs compteurs de concentration de CO_2 dans Elektor. Presque tous sont dotés d'une interface Wi-Fi qui permet de vérifier les relevés depuis n'importe où dans le monde. Ce feu tricolore pour CO_2 est différent ; il se connecte à l'IdO en utilisant le réseau Sigfox. Cela lui confère une portée nettement plus grande et lui permet de fonctionner sans accès à un réseau Wi-Fi.

Diverses études ont montré que la qualité de l'air diminue dans les espaces publics non ventilés ; une forte concentration de CO_2 est corrélée à une charge virale plus élevée dans l'air que nous respirons. Il est important de maintenir une bonne circulation de l'air afin de réduire le risque de transmission des virus en suspension dans l'air. Il existe de nombreux modules avec capteurs de CO_2 et on peut les utiliser avec un microcontrôleur pour surveiller en permanence l'air que nous respirons. Des avertissements et des messages d'alarme peuvent alors être émis lorsque le niveau de concentration mesuré dépasse un seuil prédéfini.

C'est la tâche d'un système standard de feu tricolore qui mesure le taux de CO_2. Il émet une alarme visuelle et parfois sonore lorsqu'il faut plus d'air frais. La concentration de CO_2 est affichée sur un afficheur à LED avec un feu tricolore (rouge, jaune ou vert). À l'ère de l'IdO, la plupart de ces systèmes disposent également d'une interface Wi-Fi afin que les mesures puissent être émises vers une plate-forme du nuage et affichées sur une page web consultable partout dans le monde.

Le système de feu tricolore pour CO_2 décrit ici se connecte à l'internet en utilisant le réseau radio Sigfox plutôt que le Wi-Fi. Sigfox est particulièrement bien adapté à cette application car nous avons besoin de n'envoyer que de petites quantités de données, et l'excellente couverture radio que procure Sigfox nous donne une flexibilité maximale pour l'emplacement du capteur. Le réseau Sigfox ne nécessite généralement qu'une poignée de stations de base pour couvrir une ville entière et la couverture est déjà très bonne dans de nombreux pays. Le système de feu tricolore pour CO_2 est ainsi idéal pour fonctionner là où l'accès à un réseau Wi-Fi n'est pas disponible.

Le matériel

Construire le feu tricolore pour CO_2 est assez facile ; on trouve aujourd'hui dans la gamme Arduino de nombreuses cartes à microcontrôleur très puissantes, utilisables pour créer ce feu tricolore pour CO_2. L'une d'elles, qui inclut la fonction de communication Sigfox, est la carte Arduino MKR FOX1200, déjà présentée dans la série d'articles Elektor « l'Internet des Objets et le renard » [5].

Si vous n'avez pas besoin de la connectivité IdO et que vous souhaitez juste afficher les mesures de CO_2 localement, vous pouvez utiliser un Arduino Uno standard qui se branche également sur la carte mère présentée ici. La **figure 1** présente le schéma du circuit avec la connexion à la carte mère de IC3 (Arduino Uno R3) ou IC1 (Arduino MKR FOX1200).

La carte à microcontrôleur enregistre les valeurs mesurées (concentration de CO_2, température et humidité de l'air) par le capteur de CO_2 SCD30 à intervalle régulier (réglable) et les affiche sur un écran OLED de 3,3 cm. Une matrice à LED multicolore NeoPixel constitue l'affichage du feu tricolore.

Le capteur de CO_2 SCD30 [1] utilise un bus I^2C (SDA, SCL, +5 V et GND) et se connecte via le bornier à vis à double étage, à empreinte réduite, X1. Si nécessaire, on peut connecter des résistances de rappel de 4,7 kΩ à SDA et à SCL. L'écran OLED [2] se connecte également

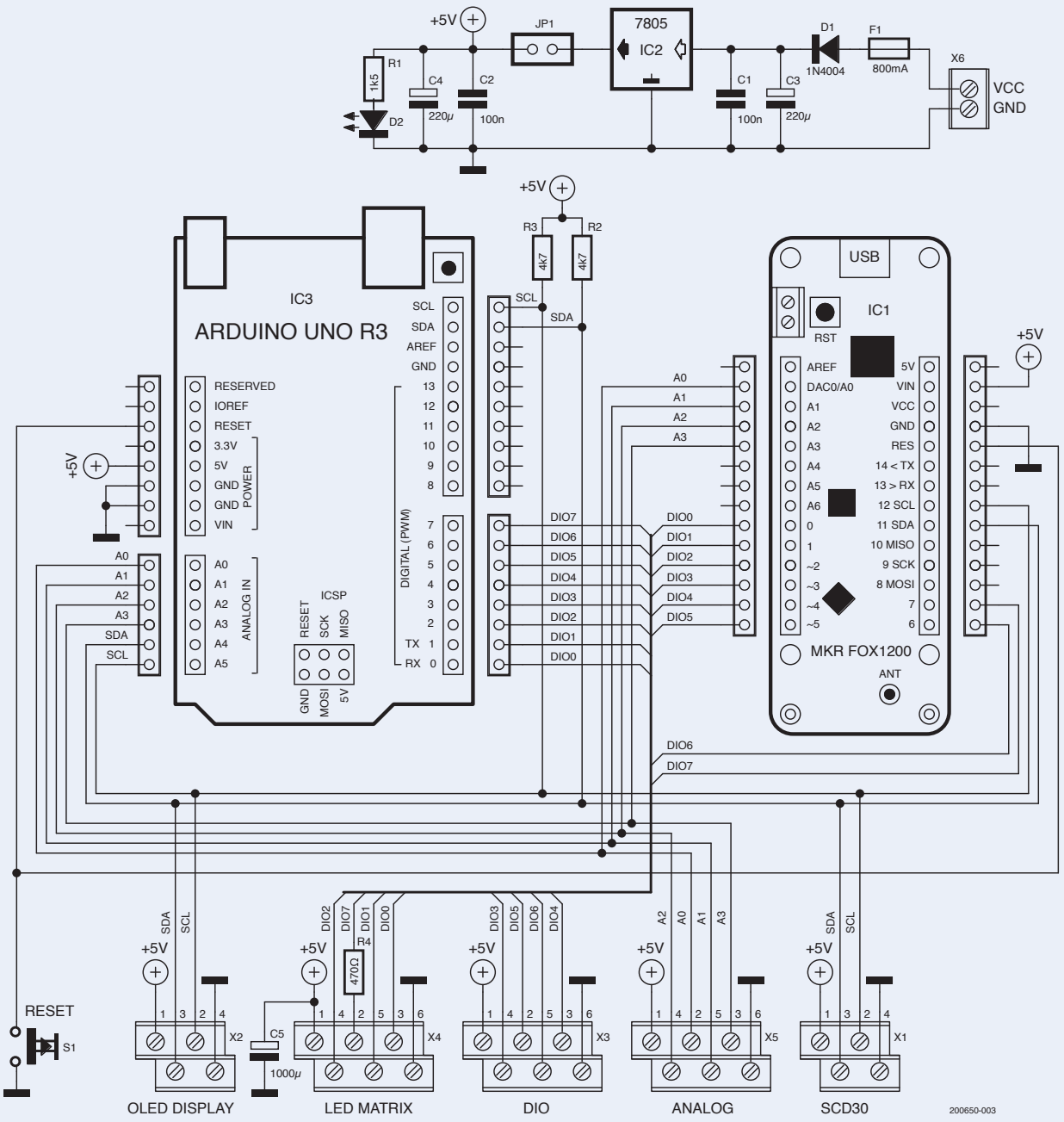

Figure 1. Le schéma du circuit du feu tricolore pour CO_2 montrant le câblage de l'Arduino Uno et de la carte MKR FOX1200.

via le bus I²C au bornier à double étage X2. En plus d'une connexion d'alimentation, l'afficheur matriciel à LED NeoPixel [4] ne nécessite qu'une seule broche numérique pour fonctionner. On utilise ici la broche DIO7 et l'afficheur se connecte via le bornier à double étage X4. Les afficheurs NeoPixel avec différents nombres de LED peuvent être pilotés à partir de cette unique broche.

On peut connecter d'autres capteurs/actionneurs aux connexions numériques/analogiques non utilisées X3, X4 et X5. Ces connecteurs peuvent être utilisés pour ajouter une nouvelle fonction au système, par ex. allumer la climatisation ou actionner la ventilation lorsque les mesures dépassent les seuils.

La **figure 2** montre le circuit imprimé assemblé sur lequel se branche (face vers le bas) la carte Arduino MKR FOX1200 ou Arduino Uno dans leurs rangées de connecteurs dédiées. **Il ne faut monter qu'une seule de ces cartes à la fois sur la carte mère. N'essayez pas de monter les deux cartes avec des connecteurs d'extension**. On peut télécharger le tracé de la carte, le plan de montage, la liste de composants et le micrologiciel Arduino sur la page Elektor du projet [3]. L'antenne plate Sigfox peut se fixer à l'aide de ruban adhésif double face. La **figure 3** montre les deux versions assemblées.

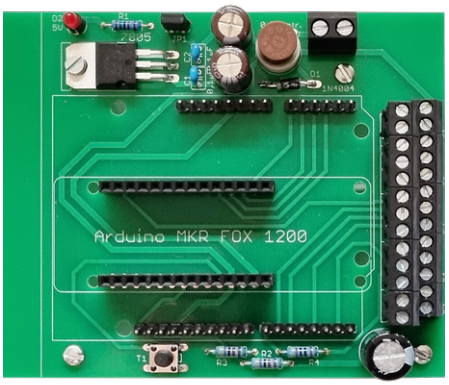

Figure 2. Le circuit imprimé de la carte mère du feu tricolore CO_2 terminé.

Figure 3a. L'Arduino MKR FOX 1200 monté sur la carte mère.

Figure 3b. L'Arduino Uno monté face à la carte mère.

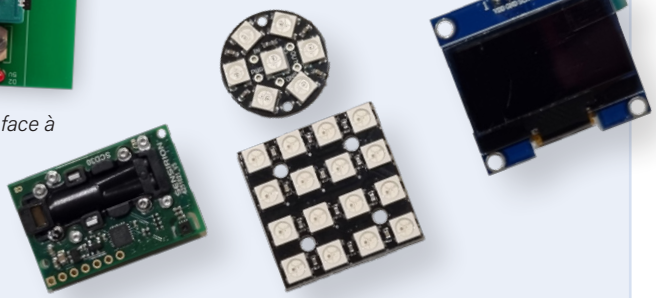

Figure 3c. Le module capteur de CO_2 (à gauche) et divers affichages.

Alimentation

Un **aspect important** de l'utilisation des NeoPixels est leur besoin en énergie. Dans le pire des cas, lorsque les trois LED individuelles d'un NeoPixel sont réglées sur l'intensité maximale avec une valeur de 255 pour produire la lumière blanche la plus brillante, chaque LED consommera 20 mA, soit 60 mA pour un seul NeoPixel. Une matrice composée de 16 NeoPixels nécessitera 16 × 60 mA = 960 mA ! C'est plus que ce que peut fournir le régulateur de tension CMS de 5 V (sans dissipateur) de la carte Arduino. L'alimenter à partir de la carte Arduino surchargerait le régulateur qui pourrait s'arrêter ou, dans le pire des cas, libérer sa fumée magique.

Dans des conditions normales de fonctionnement, vous ne solliciterez probablement pas autant les NeoPixels, mais comptez sur une consommation moyenne de 25 à 30 mA par NeoPixel, en fonction de la couleur globale affichée. Avec une matrice de 4×4 LED, cela fera entre 400 et 480 mA, ce qui est encore trop pour un régulateur de tension Arduino intégré.

Il convient par conséquent de respecter les points suivants lors de l'utilisation des afficheurs NeoPixel :

- Installez un condensateur tampon électrolytique directement sur les connexions de l'alimentation de l'afficheur NeoPixel pour atténuer les fluctuations de la tension d'alimentation lors de la commutation des LED. Utilisez un condensateur de 470 µF à 1 000 µF pour les afficheurs les plus petits. Valeur utilisée ici (C5) : 1 000 µF.
- Installez une résistance de faible valeur en série avec la ligne de commande de l'afficheur pour réduire les interférences. Valeur utilisée ici (R4) : 470 Ω.
- Une source d'alimentation externe régulée sera nécessaire, avec un régulateur de tension de haute qualité (et même refroidi) et avec des condensateurs tampons appropriés (ici nous utilisons sur la carte mère un régulateur linéaire 7805 conventionnel, alimenté par un adaptateur secteur 9 V/1 A). Les régulateurs à découpage DC/DC peuvent aussi convenir ; ils offrent un meilleur rendement et ne sont pas beaucoup plus chers.
- Il ne faut en aucun cas utiliser le régulateur de tension CMS intégré non refroidi des cartes Arduino, car des défaillances peuvent rapidement se produire, en particulier si vous utilisez une carte clonée, en raison d'une surchauffe ou de pics de courant de commutation.
- La matrice de l'afficheur à LED NeoPixel ne doit pas être plus grande que nécessaire ni être utilisée à la luminosité maximale. La luminosité de l'afficheur rond à 7 éléments NeoPixel utilisé ici est commandée par le logiciel à l'aide d'une gamme de valeurs allant de 4 à 16 sur 255 niveaux possibles.

Si le système est utilisé sans afficheur NeoPixel à des fins de test ou autres, et si la consommation totale n'est pas trop élevée, l'alimentation peut facilement être fournie par le régulateur de tension CMS intégré de l'Arduino. Dans ce cas, le cavalier JP1 sur la carte mère doit être enlevé afin que le 7805 ne soit pas alimenté à l'envers.

Ajouter quelques bibliothèques à l'EDI Arduino

Le logiciel du feu tricolore pour CO_2 nécessite quelques bibliothèques supplémentaires pour exploiter les composants externes tels que le capteur SCD30 de CO_2, l'écran OLED de 3,3 cm et la matrice de LED NeoPixel. Il faut également deux autres bibliothèques pour le fonctionnement de Sigfox avec la carte MKR FOX1200 et une bibliothèque optionnelle pour la RTC (horloge en temps réel) intégrée du microcontrôleur SAMD21.

Bibliothèque du capteur de CO_2 SCD30

Pour installer la bibliothèque de pilotes SparkFun pour le capteur SCD30, allez dans *Outils → Gérer les bibliothèques...* dans l'EDI Arduino, puis cherchez *scd30* dans le coin supérieur droit. Les deux bibliothèques SCD30 les plus courantes seront listées, une d'Adafruit et une de SparkFun. Comme le montre la **figure 4**, nous utiliserons la bibliothèque de SparkFun.

Après l'installation, la bibliothèque est automatiquement intégrée à notre programme dans l'EDI via *Croquis → Inclure une bibliothèque → ... SparkFun SCD30 Arduino Library*. Vous pouvez obtenir une description détaillée de la bibliothèque en cliquant sur *More info* en bas à gauche (**fig. 4**).

Bibliothèque de l'écran OLED de 3,3 cm

On utilise la bibliothèque *U8g2* extrêmement puissante et complète, élaborée par Oli Kraus pour piloter l'écran OLED de 3,3 cm et 128×64 pixels. Ce logiciel complet se compose en fait de quatre bibliothèques individuelles :

- *U8g2* : bibliothèque pour les applications d'affichage graphique qui utilise un grand nombre de fonctions graphiques et de jeux de caractères.
- *U8x8* : bibliothèque pour les applications d'affichage simple, purement textuel, avec une gamme réduite de jeux de caractères.
- *MUIU8g2* : fonctions spéciales pour la réalisation d'interfaces utilisateur monochromes interactives et graphiques (MUI).
- *U8log* : fonctions d'émulation d'un terminal, similaires à la fonction de moniteur série de l'EDI Arduino.

La bibliothèque du pilote U8g2 pour l'écran OLED s'installe aussi dans l'EDI Arduino en utilisant *Outils* → *Gérer les bibliothèques…* ; recherchez ensuite *u8g2* dans le coin supérieur droit (**fig. 5**).
Après l'installation, les trois bibliothèques de base sont automatiquement intégrées dans notre programme à l'aide de *Croquis* → *Inclure une bibliothèque* → *… U8g2*.

```
#include <MUIU8g2.h> // supprimer
#include <U8g2lib.h> // supprimer
#include <U8x8lib.h>
```

Nous n'avons besoin que de la bibliothèque textuelle *U8x8lib*, donc les déclarations `#include` faisant référence aux deux autres bibliothèques peuvent être supprimées.
L'avantage du pack de bibliothèques U8g est, entre autres, le grand nombre de types d'affichage pris en charge, notamment :

- Écrans à capacité graphique LCD, OLED
- Diverses résolutions de pixels
- Commande par différents bus : I²C/SPI en implémentation matérielle ou logicielle, bus parallèles selon la spécification 8080 et 6800
- Divers pilotes

Même s'il est indéniable que la surabondance d'écrans pris en charge ne nuit pas, veillez à choisir le bon, sinon vous rencontrerez des problèmes. Dans la **figure 6,** vous pouvez voir en commentaires une partie seulement de la myriade de types d'écrans possibles pris en charge.
Commencez par examiner l'un des exemples complets qui ont été installés avec la bibliothèque dans l'EDI Arduino – comme l'exemple *HelloWorld* qu'on trouve sous *Fichiers* → *Exemples* → *U8g2* → *U8x8*. Faites attention à ne sélectionner que des exemples pour la bibliothèque U8x8 !

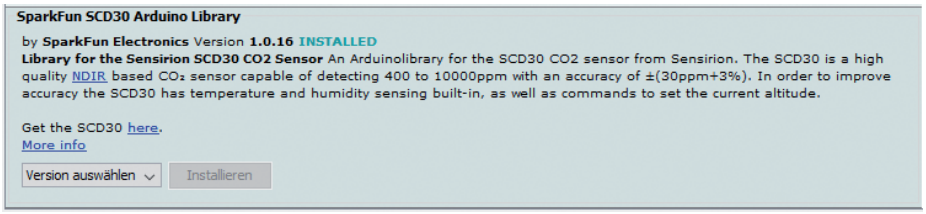

Figure 4. Sélectionnez la bibliothèque SCD30 de SparkFun.

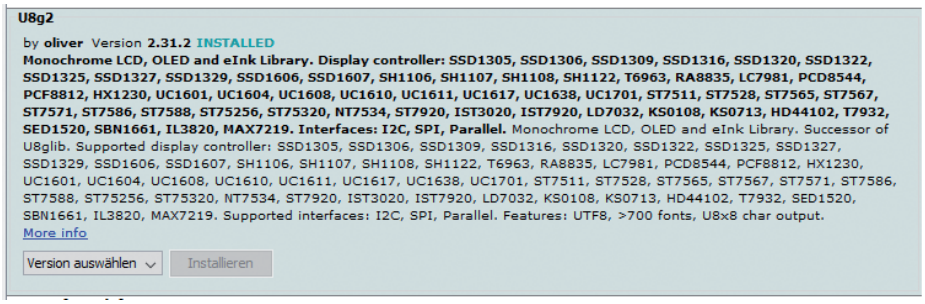

Figure 5. Sélectionnez la bibliothèque U8g2 pour piloter l'écran OLED de 3,3 cm.

Figure 6. Décommentez le bon parmi la myriade d'affichages supportés par l'U8x8.

Faites attention à ne sélectionner que des exemples pour la bibliothèque U8x8 !
Dans la longue liste de ce croquis *HelloWorld*, nous devons rechercher l'écran (OLED) utilisé. Vous l'activez en supprimant le délimiteur de commentaire `//`. Nous copions simplement la ligne 64 correspondante (commentée) du constructeur d'écran approprié à la ligne 119 [3] de notre croquis de feu tricolore CO_2 :

```
U8X8_SH1106_128X64_NONAME_HW_I2C
u8x8(/* reset=*/ U8X8_PIN_NONE) ;
```

qui se décompose ainsi :

- Bibliothèque des pilotes : `U8X8`
- Contrôleur d'écran : `SH1106`
- Nombre de pixels d'affichage : `128*64`
- Fabricant de l'écran : `NONAME` (nous utilisons un écran générique sans marque)
- Type d'interface matérielle : I²C
- Nom de l'objet créé pour référence dans le programme : `u8x8`
- Expression entre parenthèses : la broche de réinitialisation de l'affichage n'est pas utilisée.

Vous pouvez obtenir une description détaillée de la bibliothèque en cliquant sur *More info* dans le coin inférieur gauche (**fig. 5**).

Bibliothèque des LED NeoPixel

Le concept de LED NeoPixel développé par Adafruit permet d'assembler plusieurs éléments à LED RVB sous forme de rubans,

cercles et autres motifs. Les éléments à LED sont connectés en cascade et tous commandés par une seule ligne d'entrée/sortie numérique. Le pilote de LED WS2812 peut commander individuellement chacune des trois LED (rouge, vert, bleu) avec une intensité programmable sur 8 bits, ce qui permet de réaliser presque toutes les combinaisons de couleurs.

L'avantage de créer un logiciel pour les applications NeoPixel de toute taille est qu'il existe une bibliothèque Arduino très pratique d'Adafruit pour en assurer le pilotage. Cette bibliothèque est également incluse dans le paquetage du programme Arduino et est installée à l'aide du gestionnaire de bibliothèques que vous connaissez déjà (recherchez *neopixel*), comme le montre la **figure 7**. Après l'installation, la bibliothèque est automatiquement intégrée dans notre programme via *Croquis → Inclure une bibliothèque → Adafruit NeoPixel.* Ici aussi, vous pouvez trouver une description détaillée de la bibliothèque en cliquant sur le lien *More info*.

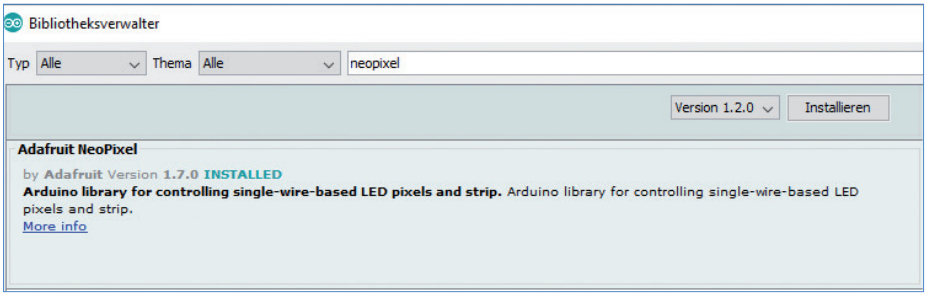

Figure 7. Installez la bibliothèque NeoPixel d'Adafruit.

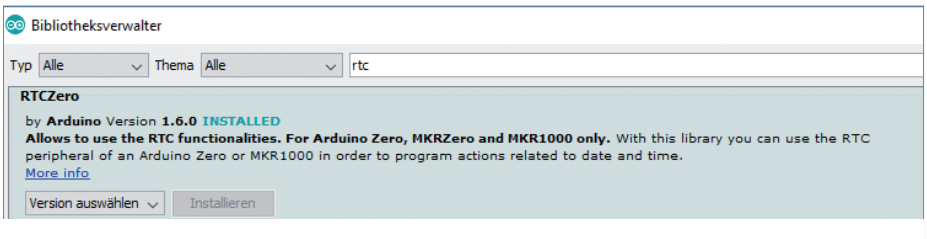

Figure 8. Sélectionnez la bibliothèque RTC pour le microcontrôleur SAMD21.

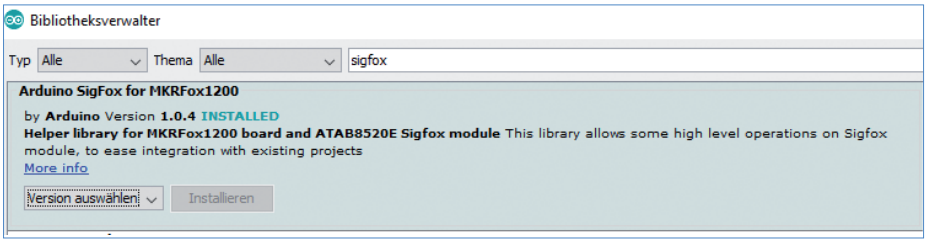

Figure 9. Sélectionnez la bibliothèque Sigfox pour la carte à microcontrôleur MKR FOX1200.

Bibliothèque RTC du microcontrôleur SAMD

Le microcontrôleur SAMD21 de la carte MKR FOX1200 possède une horloge en temps réel (RTC) intégrée, utilisable pour des applications d'heure et de date à l'échelle du système. Il faut une batterie de secours pour la carte afin que la RTC ne perde pas l'heure chaque fois que l'alimentation est coupée. Nous avons choisi de ne pas l'inclure dans le feu tricolore pour CO_2, mais un petit circuit de ce type peut être facilement connecté à la carte MKR FOX1200.

Si vous souhaitez utiliser cette RTC, il existe bien sûr une bibliothèque de pilotes adaptée, qui s'installe comme d'habitude via le gestionnaire de bibliothèques (recherchez *rtc*, **fig. 8**). Sa description détaillée se trouve aussi dans la rubrique *More info*. Elle s'intègre en utilisant *Croquis → Inclure une bibliothèque → RTCZero*.

Bibliothèque Sigfox pour MKR FOX1200

Nous en arrivons maintenant à la connexion du feu tricolore pour CO_2 à l'Internet des Objets à l'aide de la carte à microcontrôleur Arduino MKR FOX1200, qui communique avec le réseau Sigfox. Pour plus d'informations sur Sigfox, jetez un coup d'œil à l'introduction en quatre parties dans *Elektor* [5].

Grâce à Sigfox, les valeurs mesurées par le système à feu tricolore pour CO_2 sont transférées sans fil sur l'internet où elles peuvent être consultées et affichées partout dans le monde à l'aide d'un PC, d'un ordinateur portable, d'une tablette ou d'un ordiphone.

Ici encore, nous pouvons profiter de toutes les fonctions prêtes à l'emploi de la bibliothèque Sigfox disponible dans le paquetage du programme Arduino. La bibliothèque est installée via le gestionnaire de bibliothèques, comme nous l'avons fait pour les autres (recherchez *sigfox*, **fig. 9**). Installez-la maintenant avec *Croquis → Inclure une bibliothèque → Arduino Sigfox for MKRFox1200*. Cliquez sur *More info* pour en savoir plus sur les fonctions Sigfox disponibles.

Des extraits significatifs du programme Arduino annoté (pour la carte MKR FOX1200) sont donnés ici dans le **listage 1**. Le programme complet en allemand et en anglais [3] est documenté en détail et contient encore quelques routines de sortie série insérées à plusieurs endroits à des fins de débogage (pour le moniteur série dans l'EDI Arduino). Une fois satisfait et que tout fonctionne comme il se doit, ces sections de code peuvent être mises en commentaire ou supprimées.

Vous pouvez attribuer librement des seuils pour les niveaux de concentration de CO_2 mesurés et la couleur de feu tricolore associée. Les seuils que nous suggérons sont les suivants :

- 0 à 1000 ppm : vert saturé
- 1001 à 2000 ppm : jaune
- 2001 à 5000 ppm : orange
- 5001 à 12000 ppm : rouge saturé.

Fonctionnement avec le réseau Sigfox

Dans [5], nous avons montré en détail comment le paramétrage et le fonctionnement du module MKR FOX1200 sont mis en œuvre

Listage 1. Extraits du croquis Arduino

```
void loop()
{
  nsigned char i ;
  char text[15] ;

  // boucle
  while(1)
  {

    /* Le capteur SCD30 mesure automatiquement toutes les 2 s.
       Si les données sont disponibles, alors les lire et les stocker dans les variables*/

    if (AirSensor.dataAvailable())      // Vérifie si de nouvelles données sont disponibles
    {
      // Si oui, alors lire les 3 valeurs mesurées
      co2 = AirSensor.getCO2() ;
      temperatur = AirSensor.getTemperature() ;
      luftfeuchte = AirSensor.getHumidity() ;

      // Incrémenter le compteur
      mess_zae = mess_zae + 1 ;

      // Imprimer les valeurs sur le moniteur série
      Serial.print("N° de mesure : ") ; Serial.print(mess_zae) ; Serial.print(" // ") ;

      // Imprimer l'heure
      print2digits(rtc.getHours()) ;
      Serial.print(" :") ;
      print2digits(rtc.getMinutes()) ;
      Serial.print(" :") ;
      print2digits(rtc.getSeconds()) ;
      Serial.println() ;

      // Imprimer les mesures
      Serial.println("CO2 :      " + String(co2) + " ppm") ;
      Serial.println("Temp : " + String(temperatur-temp_cor) + " °C") ;
      Serial.println("Hum :  " + String(luftfeuchte) + " %rh") ;
      Serial.println() ;

      // Traitement de la valeur de CO2 pour le 'moniteur LED' : afficheur NeoPixel
      switch(co2)
      {
        case 0 ... 1000 : pixels.fill(gruen_satt,0,7) ;
                          pixels.show() ;
                          break ;

        case 1001 ... 2000 : pixels.fill(gelb_1,0,7) ;
                             pixels.show() ;
                             break ;

        case 2001 ... 5000 : pixels.fill(orange_1,0,7) ;
                             pixels.show() ;
                             break ;

        case 5001 ... 12000 : pixels.fill(rot_satt,0,7) ;
                              pixels.show() ;
                              break ;
      }

      // Sortie des valeurs mesurées sur l'écran OLED
      // Sortie : valeur du CO2
      u8x8.setCursor(6,2) ;
      sprintf(text,"%5d",co2) ; // sprintf ne fonctionne qu'avec les entiers
      u8x8.print(text) ;

      // Effacer les fantômes
      u8x8.setCursor(11,2) ;
      u8x8.print(" ppm ") ;
```

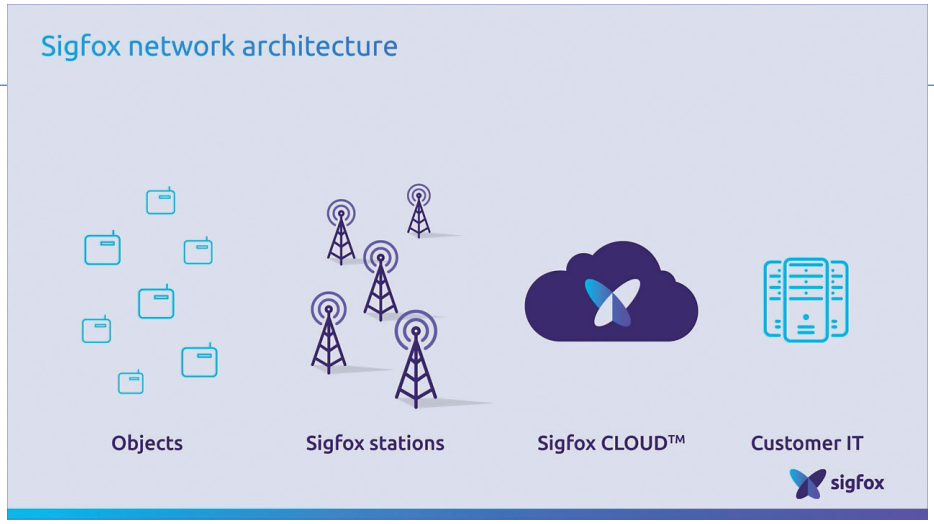

Figure 10. Structure de base du réseau Sigfox (source : Sigfox).

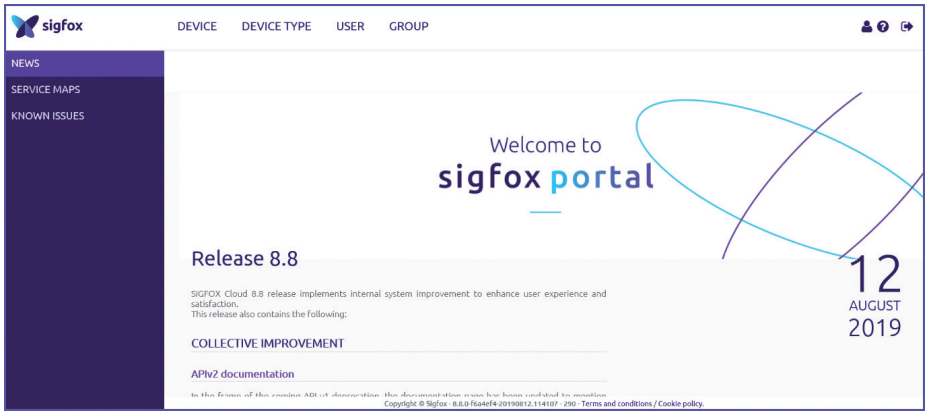

Figure 11. Le portail Sigfox (« Sigfox Backend »).

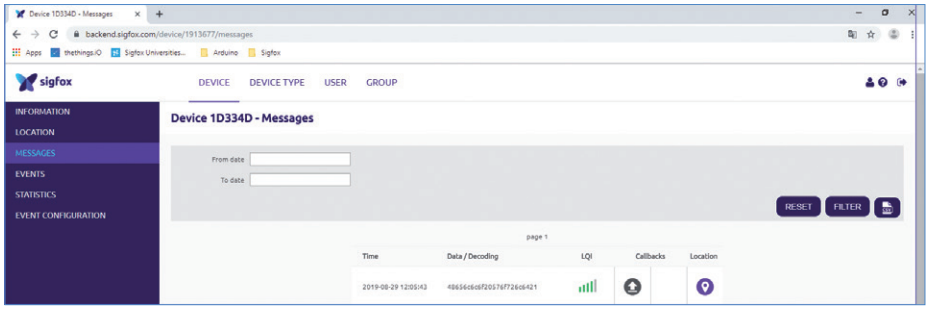

Figure 12. La fenêtre de message du périphérique.

dans le réseau Sigfox. Ici, nous pouvons nous limiter à une brève description du principe et des étapes nécessaires. Commençons par une vue d'ensemble du réseau Sigfox (**fig. 10**). Les *objets Sigfox*, comme notre appareil de mesure du CO_2 ou toute autre carte avec des capteurs, envoient leurs télégrammes avec une diffusion selon le principe « tire et oublie » sur la bande ISM sans licence 868 MHz. En plus d'une identification (ID de l'expéditeur), chaque télégramme contient un champ de données utilisateur d'une taille maximale de 12 octets, appelé charge utile. À chaque transmission, un utilisateur peut transmettre un maximum de 12 octets de données de mesure, d'état ou autres. Étant donné que le réseau Sigfox fonctionne dans la bande ISM sans licence, pour rester dans le cadre des réglementations légales, chaque objet n'est autorisé qu'à effectuer un maximum de 140 transmissions par jour. Notre carte MKR FOX1200 est donc autorisée à envoyer un télégramme toutes les onze minutes en moyenne.

En fonction de la couverture, ces transmissions sont ensuite reçues par toutes les stations de base Sigfox (*Sigfox Stations*) à portée. Ces stations de base transmettent toutes les données reçues via une connexion à l'internet ou GSM au Sigfox Cloud, à partir duquel l'utilisateur peut récupérer ses données et les exploiter dans son application de traitement des données (*Customer IT*).

L'interface de configuration du compte utilisateur est appelée *Sigfox Backend*. C'est ici que les *objets Sigfox* (appareils) sont enregistrés, que les groupes sont assignés et que le transfert de données vers le *Customer IT* (via des fonctions de rappel) est configuré.

Tout d'abord, les identifiants uniques du dispositif Sigfox *ID* et *PAC* doivent être lus à partir du dispositif Sigfox en utilisant un petit croquis Arduino [3]. Ces deux paramètres sont nécessaires pour enregistrer le dispositif Sigfox dans le Sigfox Cloud [6].

Pendant le fonctionnement, il est possible d'accéder aux télégrammes depuis le monde entier. Vous vous connectez à *Sigfox Backend* [7] avec votre adresse de courriel et le mot de passe choisi, ce qui vous amène à la page d'accueil du portail (**fig. 11**). Vous cliquez sur l'onglet *Device* pour faire apparaître une liste des appareils Sigfox actifs. Cliquez ensuite sur le champ *Id* d'un périphérique, pour accéder à la page d'information de ce périphérique. Cliquer sur *MESSAGES* sur le côté gauche active la fenêtre dans laquelle sont listés tous les télégrammes que Sigfox a reçus de ce périphérique (**fig. 12**).

Création de fonctions de rappel dans *Sigfox Backend*

Ces données brutes ne sont pas particulièrement significatives. Dès que le MKR FOX1200 envoie des données à *Sigfox Backend*, nous voulons qu'elles soient automatiquement transmises au programme de tableau de bord *Thinger.io*. *Sigfox Backend* propose à cet effet des *Callbacks* (fonctions de rappel).

Une telle fonction n'est rien d'autre qu'un transfert automatique vers le destinataire souhaité, qui se produit immédiatement lorsque *Sigfox Backend* reçoit des données d'un appareil Sigfox – par ex. de notre module MKR FOX1200. La création et la configuration d'une fonction de rappel ont été traitées en détail dans la troisième partie de la série d'articles [5], aussi nous limiterons-nous ici à l'essentiel.

Pour créer une fonction de rappel, cliquez sur l'onglet *Device Type* sur la page principale de

```
// Sortie de la valeur de la température = nombre flottant
               oled_float(6,4,temperatur-temp_cor,1) ;

      // Sortie de la valeur de l'humidité = nombre flottant
      oled_float(6,6,luftfeuchte,1) ;

      // Temps d'attente pour l'acquisition des valeurs mesurées et la transmission du télégramme Sigfox
      delay(w_zeit * 60000) ; // (w_zeit * 1 minute) attente entre les mesures
      min_zae = min_zae + w_zeit ; // Décompte des minutes jusqu'à la prochaine transmission Sigfox
      if (min_zae == SF_zyk) // Maintenant : envoyer le télégramme Sigfox
      {
        SF_send_data() ;
        min_zae = 0 ; // Remise à zéro du compteur
} } } }

/*** Envoyer des données via Sigfox ***/

void SF_send_data(void)
{
  Serial.print("Sigfox - Start ... \n") ;

  // Ecrire les valeurs mesurées dans la variable de la structure de données
  SF_Ampel.CO2 = co2 ;
  SF_Ampel.Temp = temperatur ;
  SF_Ampel.Feucht = luftfeuchte ;


  // Si nécessaire : sorties de débogage
/* Serial.println() ;
  Serial.print("CO2 : ") ; Serial.println(SF_Ampel.CO2) ;
  Serial.print("Temp : ") ; Serial.println(SF_Ampel.Temp) ;
  Serial.print("Feucht : ") ; Serial.println(SF_Ampel.Feucht) ;
  Serial.println() ;
*/

  // Activer le modem Sigfox et interroger les erreurs.
  if (!SigFox.begin())      // Initialisation du modem
  {
    Serial.println("Modem Sigfox introuvable ! - Continuer avec RESET !") ;
    while (1) ; // boucle
  }
  else
  {
    Serial.println("Initialisation du modem Sigfox OK !") ;
  }

  // Activer la LED de débogage et désactiver les modes de veille.
  SigFox.debug() ;

  // Effacer toutes les interruptions en attente.
  SigFox.status() ;

  // Envoyer la charge utile via Sigfox
  SigFox.beginPacket() ; // Préparation de l'envoi d'un paquet.

  // Envoyer la variable de structure au Sigfox Backend
  SigFox.write((char*)&SF_Ampel, sizeof(SF_Ampel)) ;

  // Vérification des erreurs - si endPacket() renvoie 'true' : erreur
  SF_error = SigFox.endPacket()
  if(SF_erreur > 0)
  {
    Serial.println("Erreur Sigfox !!") ;
  }

  // Fin de Sigfox
  SigFox.end() ;

  Serial.println("Sigfox - Fin .... !\n") ;
}
```

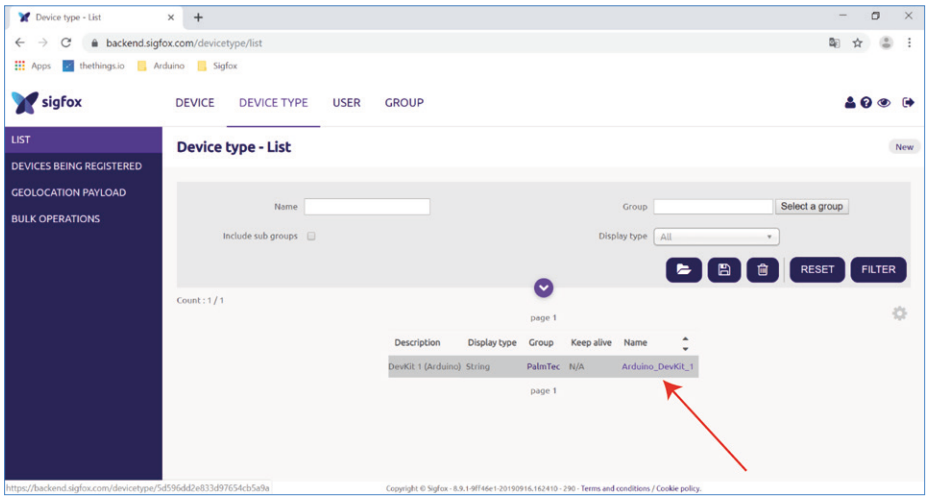

Figure 13. Choisissez le type de dispositif souhaité.

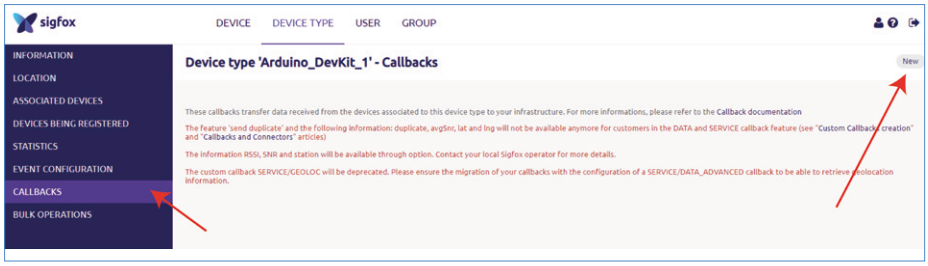

Figure 14. La fenêtre « Callback ».

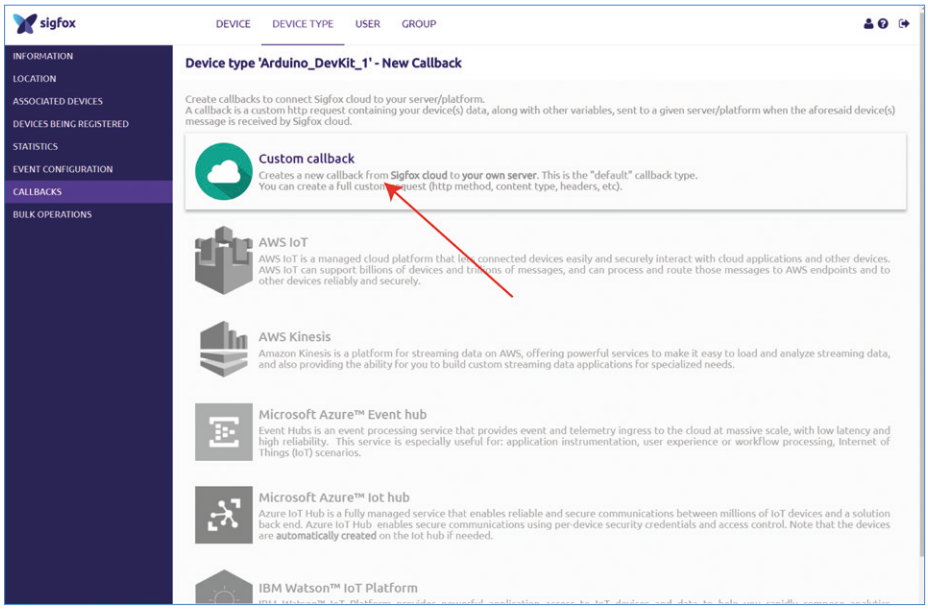

Figure 15. Le large choix de types de fonctions de rappel.

Sigfox Backend et dans la liste qui apparaît (avec une seule entrée) sur le nom du *Device Type*, dans notre cas *Arduino_DevKit_1* (**fig. 13**). Dans la page d'aperçu du *Device Type*, cliquez sur *Callbacks* dans la liste à gauche (**fig. 14**) et dans cette fenêtre en haut à droite, cliquez sur le petit bouton *New*. Une liste apparaît avec toutes sortes de rappels possibles pour les types les plus courants de programmes de tableaux de bord ou du nuage (**fig. 15**). Ici, vous sélectionnez *Custom Callback*.

Dans la fenêtre qui apparaît maintenant (**fig. 16**), se configure la fonction de rappel pour l'envoi des données de *Sigfox Backend* à *Thinger.io*.
Avec cette fonction, *Sigfox Backend* envoie en permanence au tableau de bord, dès réception du télégramme du feu tricolore CO_2, les variables `Device-ID`, `Telegr-Nr`, `CO2`, `Temp` et `Feucht` (humidité) avec leurs valeurs.

Créer un tableau de bord avec Thinger.io

Thinger.io est une plateforme de visualisation IdO *open source* qui permet de créer rapidement et facilement une présentation claire et détaillée des données. Elle est gratuite pour les petits projets. Les valeurs sont visualisées sur un tableau de bord, qui peut aussi être accessible publiquement aux navigateurs web. Il suffit de quelques étapes simples pour développer un *tableau de bord* personnalisé :

- Créez un *compte* utilisateur gratuit sur thinger.io.
- Créez un conteneur de données (*Data Bucket*) chez thinger.io pour contenir les valeurs mesurées envoyées par le cloud Sigfox.
- Définissez le point d'accès (*Token*) pour le Sigfox Cloud et créez une authentification de réception (*Access Token*) chez thinger.io, afin que le Sigfox Cloud reçoive la permission de transmettre des données à thinger.io.
- Configurez une fonction de rappel dans le Sigfox Cloud pour transférer les données du Sigfox Cloud vers le *Data Bucket* de thinger.io via internet.
- Concevez un tableau de bord soigné sur le site thinger.io pour visualiser les données.

Ici aussi, nous vous renvoyons à [5], qui donne une description plus détaillée des différentes étapes. Après avoir créé un compte gratuit sur thinger.io [8], vous vous connectez et accédez à l'écran principal, qui est le point de départ de toutes les actions ultérieures (**fig. 17**).
Pour créer un tableau de bord, cliquez sur *Dashboards* à gauche, puis sur le bouton *Add Dashboard* dans la fenêtre qui s'affiche. Un grand nombre de *widgets* (les composants de l'interface utilisateur) librement configurables sont disponibles pour la conception personnalisée d'un tableau de bord, par ex. des

graphiques montrant les valeurs de mesure dans le temps, des graphiques en anneau, des bargraphes, des affichages analogiques, Google Maps pour montrer la localisation, des images, du texte, des LED et une horloge.
La **figure 18** montre un exemple de tableau de bord conçu avec thinger.io pour notre feu tricolore CO_2. Les valeurs mesurées sont affichées sous la forme d'un diagramme temporel. En plus de ces données, il est possible d'afficher l'heure, le numéro de télégramme et d'autres informations. Un accès mondial au tableau de bord est possible en quelques clics de souris, de sorte que toute personne qui reçoit le lien peut ensuite lire le tableau de bord et vérifier les valeurs courantes.

200650-04 – VF : Denis Lafourcade

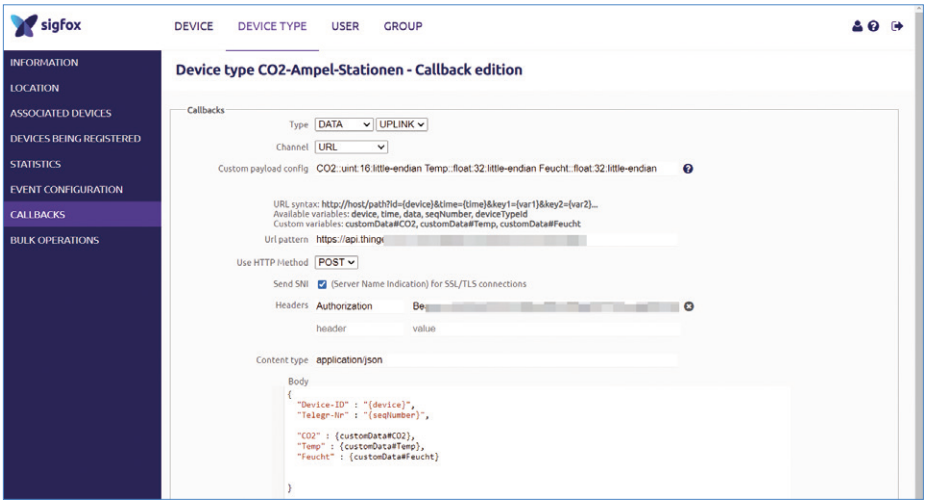

Figure 16. Configuration de la fonction de rappel.

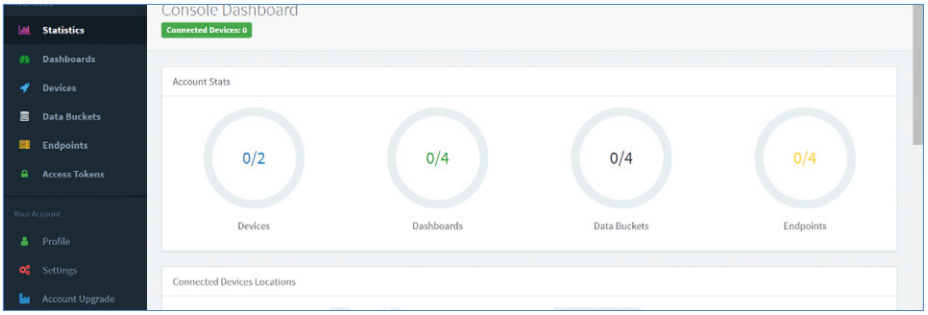

Figure 17. Statistiques du compte sur le tableau de bord de la console.

PRODUITS

- **Arduino MKR FOX 1200**
  www.elektor.fr/19096
- **Arduino Uno Rev3**
  www.elektor.fr/15877
- **Seeed Studio Grove SCD30 CO_2, capteur de température et d'humidité pour Arduino**
  www.elektor.fr/20012

Figure 18. Le tableau de bord du feu tricolore CO_2.

LIENS

[1] Capteur de CO2 SCD30 : https://bit.ly/34XbL5o
[2] Écran OLED de 3,3 cm : https://bit.ly/3fEP7AX
[3] Page Elektor du projet : https://www.elektormagazine.fr/200650-04
[4] Afficheurs à LED NeoPixel (article en allemand) : https://bit.ly/3qf1Y2k
[5] « l'Internet des Objets et le renard (1) - une carte abordable pour accéder au réseau Sigfox » (et articles suivants), Frank Schleking et Bernd vom Berg, Elektor 11-12/2019 : http://www.elektormagazine.fr/190281-04
[6] Enregistrement d'un dispositif Sigfox dans le Sigfox Cloud : https://buy.sigfox.com/activate/devkit/FR
[7] Page de connexion de Sigfox Backend : https://backend.sigfox.com/auth/login
[8] Tableau de bord sur thinger.io : http://www.thinger.io